AMENDMENTS TO THE SPECIFICATION

The following amendments are with respect to the paragraph numbering of U.S. Patent Publication US 2005/0015758.

Please amend paragraph [0028] as follows:

[0028] FIG. 1 illustrates a target processor 13 including target registers 15 together with memory 18 storing a number of software components 19, 20, 21, and providing working storage including a basic block cache 23, a global register store 27, and the subject code 17 to be translated. The software components include an operating system 20, the translator code 19, and translated code 21. The translator code 19 may function, for example, as an emulator translating subject code of one ISA into translated code of another ISA or as an accelerator for translating subject code into translated code, each of the same ISA.

Please amend paragraph [0044] as follows:

[0044] FIG. 2 shows an example of translation and execution interlaced. The translator 19 first generates translated code 21 based on the subject instructions 17 of a first basic block 153, then the translated code for basic block 153 is executed. At the end of the first basic block 153, the translated code 21 returns control to the translator 19, which then translates a second basic block 159, including generating a respective IR tree 165 with appropriate abstract registers 181, 183, 185, 187, and instruction elements 189, 191, 193 and 195. The translated code 21 for the second basic block ~~161~~ 159 is then executed, at step 157. At the end of the execution of the second basic block 159, the translated code returns control to the translator 19, which then translates the next basic block, and so forth.

Please amend paragraph [0154] as follows:

[0154] FIG. 8 illustrates the steps performed by the translator at run-time, between executions of translated code. When a first basic block (BB_{N-1}) finishes execution 1201, it returns control to the translator 1202. The translator increments the profiling metric of the first basic block 1203. The translator then queries the basic block cache 1205 for previously translated isoblocks of the current basic block (BB_N, which is BB_{N-1}'s successor), using the subject address returned by the first basic block's execution. If the successor block has already been translated, the basic block cache will return one or more basic block data structures. The translator then compares the successor's profiling metric to the group block trigger threshold 1207 (this may involve aggregating the profiling metrics of multiple isoblocks). If the threshold is not met, the translator then checks if any isoblocks returned by the basic block cache are compatible with the working conditions (i.e., isoblocks with entry conditions identical to the exit conditions of BB_{N-1} at step 1209). If a compatible isoblock is found, that translation is executed 1211.

Please amend paragraph [0167] as follows:

[0167] According to the process illustrated in FIG. 10, the translator 19 translates a first code sequence CS1 into target code TC1, as illustrated in step 121. The translator 19 then generates a cache key K_1 which indexes the target code block TB1 corresponding to code sequence CS1, as illustrated in step 123. In step 124, the translator 19 stores the target block TC_1 along with its associated key K_1 into cache storage. In step 125, the translator 19 begins processing a second code sequence CS2, first generating a cache key K_2 for that sequence. Then in comparison step 127, the translator 19 compares the cache key K_2 to those keys associated with codesequences previously stored in the cache 29, including key K_1. If the cache key K_1 matches the cache key K_2, then, as illustrated at step 129, the target code block TC_1 corresponding to the cache key K_1 is retrieved from the cache 29 by the translator 19 and executed. ~~As shown in step 131, the~~ The flow then proceeds to step 133 wherein the translator 19 then begins to process code sequence CS3, first generating a cache key K_3 for that sequence and then examining the index of cache keys . . . K_1, K_2 . . . for a match. If K_1 does not match K_2 at step 127, the second code sequence is translated into target code TC_2, which is then cached, as illustrated in steps 135, 136.

Please amend paragraph [0201] as follows:

[0201] According to the illustrative process of FIG. 16, a first translator execution, step 201, is performed and then, in step 203, its cache units C_1 are cached. A second instance of the translator 19 then executes a second program, and generates cash units C_2, step 205. The second translator instance 19 then compares its translation structures C_2 at the end of its execution with those translation structures C_1 stored in the cache 29 in step 207, and determines if the just-produced translations are "better" than the ones available in the cache 29 according to some appropriate criteria. For example, determining whether one code cache is better than another may be based on the number of subject instructions translated, the number of optimizations applied, or by execution of an algorithm which evaluates the quality of the generated code. As illustrated in step 209, if C_2 is better then C_1, then cash structures C_2 are loaded into the cache 29, replacing the structures C_1. Otherwise, the structure C_1 is retained at step 211.